

## Protection and Security

CS 256/456  
Dept. of Computer Science, University  
of Rochester

11/13/2008 CSC 2/456 1

## Security and Protection

- Goals:
  - Data confidentiality
  - Data integrity
  - System availability
  - Privacy
  
- Malicious intruders
- Accidental data loss

11/13/2008 CSC 2/456 2

## Operating Systems Protection

- Operating system consists of a collection of objects, hardware or software (e.g., files, printers)
- Protection problem - ensure that each object is accessed correctly and only by those processes that are allowed to do so
  - a specific type of security problem
- Domain = set of (object, rights) pairs
  - what is a domain in traditional UNIX?
  - when do process domain switches occur in traditional UNIX?

Domain 1: File1[R], File2[RW]

Domain 2: File3[R], File4[RWX], File5[RW]

Domain 3: File6[RWX], Plotter2[W]

Intersection of 2 and 3: Printer1[W]

11/13/2008 CSC 2/456 3

## Domain Structure

- Principle of protection: least privilege
- Alternatives for domain of protection:
  - Unix: users, groups
  - Multics (1960s): rings of capabilities/protection

11/13/2008 CSC 2/456 4

### How to Track Objects/Rights in Domains?

- View protection as a matrix (*protection matrix*)
  - Rows represent domains
  - Columns represent objects
  - $Access(i, j)$  is the set of operations that a process executing in Domain<sub>i</sub> can invoke on Object<sub>j</sub>
- Dynamic protection
  - Operations to add, delete access rights

		Object							
		File1	File2	File3	File4	File5	File6	Printer1	Plotter2
Domain	1	Read	Read Write						
	2			Read	Read Write Execute	Read Write		Write	
	3						Read Write Execute	Write	Write

11/13/2008

CSC 2/456

5

### Implementation of Protection

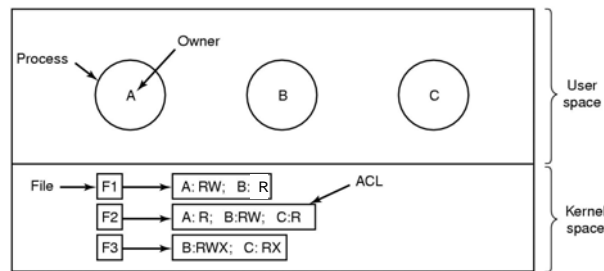
- The complete protection matrix consumes too much space and is usually very sparse.
  - two ways to condense it
- Each column = Access-control list for one object  
 Defines who can perform what operation.  
 Domain 1 = Read, Write  
 Domain 2 = Read  
 Domain 3 = Read  
 ...
- Each Row = Capability List for each domain  
 Define what operations allowed on what objects.  
 Object 1 - Read  
 Object 4 - Read, Write, Execute  
 Object 5 - Read, Write, Delete, Copy
- What does traditional UNIX do?

11/13/2008

CSC 2/456

6

### Access Control Lists



Use of access control lists to manage file access

Revocation of access rights?

11/13/2008

CSC 2/456

7

### Capabilities

- Capability for a domain
  - abstraction that indicates access rights to objects for the domain
- Compare against ACL
  - may be built-in with handle to objects (more efficient access right checking)
  - efficient mechanism to pass it around
- Important concern
  - a process should not be able to tamper with its capabilities

11/13/2008

CSC 2/456

8

## Implementing Capabilities

- Kernel-space capability list
  - user programs use handles (e.g., file descriptors) to refer to them

- Tagged architecture
  - memory words containing capabilities are tagged; user programs can only read those words; only kernel programs can change those words
  - the protection system kernel does not need to track capabilities and capabilities can be passed by memory sharing

11/13/2008 CSC 2/456 9

## Cryptographically-protected Capabilities

- Cryptographically-protected capability
  - a random number (called "check") is generated for each file at creation time and maintained in secrecy
  - capability is formed cryptographically

Server	Object	Rights	f(Object, Rights, Check)
--------	--------	--------	--------------------------

- Function f() is a one-way function such that:
  - it is computationally infeasible to guess what "check" is even when object, rights, and f(object, rights, check) is known.
- Compared with tagged capabilities, cryptographically-protected capability
  - does not require new hardware support
  - can be passed around between distributed machines

11/13/2008 CSC 2/456 10

## Revocation of Access Rights for Capabilities

- Revocation:
  - change the secret check number?
- total*: revoking rights to all processes
- selective*: revoking rights to a selected group of processes (e.g., all processes belonging to a particular user)
  - Timeout & Reacquisition

11/13/2008 CSC 2/456 11

## Protection in UNIX

- Protection domains: users
- Access matrix for files:
  - a simplified access control list
- Protection commands for files:
  - each user can change protection on files it owns
  - superuser can do everything

11/13/2008 CSC 2/456 12

### Protection in Java

- A Java class can be loaded remotely; therefore can be dangerous
  - A class is assigned a protection domain when it is loaded by the JVM
  - The protection domain indicates what operations the class can (and cannot) perform
- If a library method is invoked that performs a privileged operation, the stack is inspected for access right violation
- Other techniques
  - Sandboxing
  - interpretation

...
untrusted applet
URL loader
network library

stack top

11/13/2008
CSC 2/456
13

### Trusted Systems

- Trusted Computing Base
  - hardware and software for enforcing security rules
- If the TCB works according to specification, the system security cannot be compromised
- A reference monitor-based TCB:

11/13/2008
CSC 2/456
14

### Formal Models of Secure Systems

		Objects		
		Compiler	Mailbox 7	Secret
Eric	Read Execute			
Henry	Read Execute	Read Write		
Robert	Read Execute		Read Write	

		Objects		
		Compiler	Mailbox 7	Secret
Eric	Read Execute			
Henry	Read Execute	Read Write		
Robert	Read Execute	Read		Read Write

(a)
(b)

- Protection commands: operations that can change protection matrix
- TCB must ensure that protection commands do not move the protection matrix from an authorized state to an unauthorized state

11/13/2008
CSC 2/456
15

### Covert Channels: Information leaking

(a)

(b)

Client, server, and collaborator processes

Encapsulated server can still leak to collaborator via covert channels

- Covert channels
  - Modulating CPU usage
  - Locking/unlocking files

11/13/2008
CSC 2/456
16

## Login Spoofing

- Login spoofing
  - A program running by the attacker displays a login screen (like the real one)
  - After a legitimate user types in username and password, it records those, kills itself, and a real login screen is shown
  - The user thinks she typed in a wrong password and tries again, which works
- Countermeasure?
  - Start each login session with a non-user-catchable key combination "Ctrl-Alt-Delete"

11/13/2008

CSC 2/456

17

## Leaking Unnecessary Information

```
LOGIN: ken
PASSWORD: FooBar
SUCCESSFUL LOGIN
```

(a)

```
LOGIN: carol
INVALID LOGIN NAME
LOGIN:
```

(b)

```
LOGIN: carol
PASSWORD: Idunno
INVALID LOGIN
LOGIN:
```

(c)

User authentication:

- (a) A successful login
- (b) Login rejected after name entered
- (c) Login rejected after name and password typed

11/13/2008

CSC 2/456

18

## User Authentication

- UNIX user passwords are mapped using a one-way function "e()"; and then stored in a globally readable file "/etc/passwd"
  - Bobbie, e(Dog)
  - Tony, e(6%%TaeFF)
  - ... ..
- Attack:
  - used a precomputed common password list
- Countermeasure?
  - salt
  - Bobbie, 4238, e(Dog4238)
  - Tony, 2918, e(6%%TaeFF2918)

11/13/2008

CSC 2/456

19

## Improving Password Security

- One-time passwords
  - Challenge-response authentication
- Authentication using a physical object (e.g., ATM card)
- Authentication using biometrics

11/13/2008

CSC 2/456

20

### The TENEX Password Problem

- Files are accessed with passwords. At each access, the password is checked byte-by-byte and an error is returned as soon as a byte is mismatched.

11/13/2008

### Buffer Overflow

(a) Situation when main program is running  
 (b) After program A called  
 (c) Buffer overflow shown in gray

Countermeasures:

- boundary checks, non-executable stack/data segment, ...

11/13/2008 CSC 2/456 22

### The Morris Worm [1988]

- Three methods of infection
  - rsh from trusted machine
  - Buffer overflow attack in finger
  - Bug in sendmail
- Once a machine was infected, used password cracking for further proliferation
- If a copy of the worm already existed, 1 in 7 exited
  - Downfall since it brought the Internet down

11/13/2008 CSC 2/456 23

### The Security Environment

Security goals:

- Authentication
- Data confidentiality
- Data integrity
- System availability

Threats of intruders or adversaries:

- Identity hijacking
- Exposing data
- Tampering with data
- Denial of service attacks

We focus on OS-related security issues

11/13/2008 CSC 2/456 24

## So far ...

- Login and password security
- Cryptography as a tool
- Buffer overflow attack

11/13/2008

CSC 2/456

25

## Security Services

- Privacy (prevent others from obtaining information)
- Authentication (verify the identity of an object owner)
- Data integrity (make sure the data is not altered)

→ Cryptography as a Security Tool

11/13/2008

CSC 2/456

26

## Cryptographic Algorithms

- Encryption/decryption algorithms typically public knowledge
- Key shared between sender and receiver is the difference
  - Secret key – symmetric – both participants share a single key
  - Public key (e.g., RSA)
    - Public key published for all to know
    - Private key that is shared with no one
  - Hash or message digest (e.g., MD5) – no keys
    - Map a potentially large piece of data into a small fixed-length number
    - More efficient to compute than the above techniques

11/13/2008

CSC 2/456

27

## Encryption Algorithm Requirement

- Assumption – only the key is kept secret or you will have to keep re-inventing new encryption algorithms!
  - Algorithm must be safe from chosen plain-text attacks (if the attacker knows both plaintext and ciphertext)
    - Make sure none of the structure of plaintext remains
- Key distribution a central problem in security

11/13/2008

CSC 2/456

28

## DES – Data Encryption Standard

- Secret key algorithm
- Encrypts 64-bit blocks of plaintext using a 64-bit key
- Three distinct phases
  - Permute (shuffle bits in the block)
  - Perform 16 rounds of an identical operation
  - Perform the inverse of the original permutation
- Decryption – apply the same algorithm, except that the keys are applied in reverse

11/13/2008

CSC 2/456

29

## Encrypting Large Data

- Cipher block chaining (CBC) – XOR ciphertext for block  $i$  with plaintext for block  $i+1$  before running it through DES
- Use initialization vector in lieu of non-existent ciphertext for block 0
- Use Triple-DES to increase security

11/13/2008

CSC 2/456

30

## RSA (Rivest, Shamir, and Adleman) Algorithm

- Public key for encryption, private key for decryption
- Grounded in number theory
- Encryption/decryption function requires enormous computational power

11/13/2008

CSC 2/456

31

## Key Selection

- Key length of 512 bits
- Choose two large prime numbers  $p$  and  $q$ , and multiply them together to get  $n$
- Choose encryption key  $e$  such that  $e$  and  $(p-1)(q-1)$  are relatively prime
- Compute decryption key  $d$  such that
- Public key constructed from the pair  $\langle e, n \rangle$  and private key by the pair  $\langle d, n \rangle$

11/13/2008

CSC 2/456

32

### Authentication: Simple Three-Way Handshake

- Participants already share a secret key
- Client send encrypted (using client handshake key) random number  $x$  along with clientId
- Server decrypts and responds with  $x+1$ , along with random number  $y$ , both encrypted with server handshake key
- Client validates legitimacy of server by checking return value  $x+1$  and returns encrypted  $y+1$
- Server validates legitimacy of client based on response and then sends encrypted session key

11/13/2008

CSC 2/456

33

### Authentication: Trusted Third Party (e.g., Kerberos)

- The authentication server  $S$  is trusted and shares a secret key with  $A$  and  $B$ ,  $K_A$  and  $K_B$  respectively
- $A$  identifies  $A$  and  $B$  to  $S$
- $S$  sends back a two-part message, encoding session key  $K$  and a timestamp  $T$ , using  $K_A$  and  $K_B$
- $A$  decodes first part of message and forwards second part to  $B$  along with timestamp encoded using  $K$
- $B$  decodes second part to recover  $K$  and  $T$  and uses  $K$  to decode second part and compare
- $B$  replies with  $T+1$  encoded using  $K$

11/13/2008

CSC 2/456

34

### PGP – Pretty Good Privacy

- Provides privacy, authentication, integrity, compression
  - Hash message using MD5
  - Encrypt hash using private RSA key
  - Compress using ZIP
  - Use random key to encrypt message using IDEA (similar to DES)
  - Encrypt random key using receiver's public key

11/13/2008

CSC 2/456

35

### Information Leaking Through Side Channels

- Side channels
  - performance observations
  - program execution signals (e.g., cache usage, memory bus usage)
- Side channel attack on hyper-threading processors [Percival 2005]
  - OpenSSH running DES encryption on one hyper-thread
  - attacker running on the other hyper-thread
  - attacker and OpenSSH share hardware cache, so attacker can monitor its own cache miss pattern to infer the execution of OpenSSH (and its DES encryption key)

11/13/2008

CSC 2/456

36

## Denial-of-Service Attack

- Attacker attempts to consume all available resources at the host so no resources are left to serve legitimate users
  - attacks often come from network and they are distributed
- TCP flooding:
  - attackers establish many bogus TCP connections
  - host allocates buffer space for each connection
  - host memory being exhausted eventually
- Countermeasure?
  - **discard** flooded requests: throw out good and bad ones
  - **trace back** to source of floods
    - attack requests with spoofed identities
    - sources are most likely an innocent, compromised machines
  - **delayed processing/resource allocation**

11/13/2008 • stateless TCP [Shieh et al., NSDI2005] 37

## Virus

- Virus (fragment of code embedded in a legitimate program)
  - program can reproduce itself
    - e.g., when invoked, traverse the file system and attach it to randomly selected executables
  - additionally, do harm
    - steal your data
    - temporarily crash the system
    - permanently damage data or hardware
    - denial of service by using all available system resources
- "Good" virus
  - quickly spreading virus
  - difficult to detect
  - hard to get rid of

11/13/2008 • CSC 2/456 38

## Infecting An Executable (Trojan Horses)

(a) An executable program  
 (b) With a virus at the front  
 (c) With the virus at the end  
 (d) With a virus spread over free space within program

11/13/2008 CSC 2/456 39

## Memory Resident Viruses

- Virus resides in memory; intercepting system calls
- Where in memory to put the virus?
  - known unused memory in OS kernel
  - make the OS believe the memory that virus uses is "legitimately used"
- How to load virus there in the first place?
  - boot sector viruses
  - device driver viruses

11/13/2008 CSC 2/456 40

## How Viruses Spread

- Try to infect programs on
  - networks: exploiting buffer overflow errors in network server daemons
  - floppy drives
- Attach to innocent looking email
  - when it runs, use mailing list to replicate

11/13/2008

CSC 2/456

41

## Antivirus Techniques

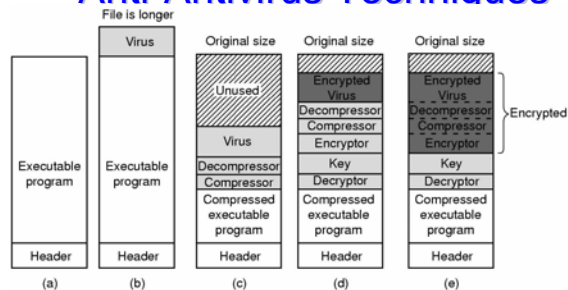
- Size checkers
  - keep a record of the size of disk files and scan them periodically for any size changes
  - apply on read-only executables
- Signature scanning
  - maintain a database of patterns of common viruses
  - scan disk files for these patterns

11/13/2008

CSC 2/456

42

## Anti-Antivirus Techniques



- (a) A program
- (b) Infected program
- (c) Compressed infected program
- (d) Encrypted virus
- (e) Compressed virus with encrypted compression code

11/13/2008

CSC 2/456

43

## More Antivirus Techniques

- Integrity checkers
  - similar to size checkers, but this time we compute a checksum for file and store them somewhere; we periodically check all files to see whether the checksum still matches
- Behavioral checkers (memory-resident anti-virus program)
  - intercept system calls and detect suspicious activities: overwriting executables, ...

11/13/2008

CSC 2/456

44

## Disclaimer

- Parts of the lecture slides contain original work by Abraham Silberschatz, Peter B. Galvin, Greg Gagne, Andrew S. Tanenbaum, and Gary Nutt. The slides are intended for the sole purpose of instruction of operating systems at the University of Rochester. All copyrighted materials belong to their original owner(s).

11/13/2008

CSC 2/456

45